Computer Science and Information Systems 13(2):593–608 DOI: 10.2298/CSIS160228017W

Routing Optimization for Server-Centric Data Center Networks

Huanzhao Wang[1,2], Kun Qian[1], ChengChen Hu[1], Che Zhang[1], and Yadong Zhou[1]

[1] Department of Computer Science and Technology, Xian Jiaotong University
710049 Xian, China
{hzhwang, chengchenhu, ydzhou}@mail.xjtu.edu.cn
qiankun11516@sina.cn
chezhang.china@gmail.com
[2] Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory
Shijiazhuang 050081, China
hzhwang@mail.xjtu.edu.cn

Abstract. Server-centric data center architecture has been proposed to provide high throughput, scalable construction and error tolerance with commodity servers and switches for cloud data centers. To fully utilize those advantages of server-centric data center, an effective routing algorithm to find high quality multiple paths in Server-centric network is needed. However, current routing algorithms cannot achieve this completely: 1) the state-of-art routing algorithms in server-centric data center just consider hop count when selecting paths; 2) traditional multi-constraint QoS routing algorithms only find one feasible path and are usually switch-oriented; 3) present multi-path algorithms cannot guarantee the performance of the founded paths. In this paper, we propose a multi-constrained routing algorithm for server-centric data centers, named Server-Centric Multi-Constrained Routing Algorithm (SCRAT). This algorithm exploits the topology features of the Server-Centric data center to decrease the algorithm complexity and returns optimal and feasible paths simultaneously. In simulations, SCRAT has a very high probability (more than 96%) to find the exact optimal path, and the cost of the optimal path found in SCRAT is about 10% less compared with path found in previous TS_MCOP. Compared with previous MPTCP, SCRAT reduces the path delay by 18% less and increase the bandwidth by 20%.

Keywords: server-centric, data center, routing optimization.

1. Introduction

In recent years, Data Centers (DC) has been widely employed to fulfill the increasingly demanding requirements for a variety of business needs [5, 9, 25]. Enterprises, service providers, and content providers rely on data and resources in their data centers to run business operations, deliver network services and distribute revenue-producing content [17, 18, 20]. Data Center Networking (DCN) is an important part of any modern data center, which must deliver high reliability and satisfied performance. However, in the current state, it is observed that the network is a bottleneck to computation [3], after careful analysis on the collected data from a large cloud service data center. Efficient routing inside a data center becomes one of the essential and challenging parts of DCN due to the following two reasons. First, the traffic exchanges among the servers in a data center dominates the traffic of a data center. A recent measurement reported the ratio of traffic volume between servers inside a data center to the traffic entering/leaving the data center to be 4:1 [11]. Second, the scale of the data center grows really fast and it is expected to hold hundreds of thousands of servers in a single data center. In order to interconnect such a large number of servers with commodity switches and servers, Server-Centric Data Center (SCDC) is proposed [13, 15]. In SCDC servers are equipped with multiple network interfaces and act not only as end hosts but also as relay switches for multi-hop communications. Although SCDC achieves architectural advantages, its routing algorithms contain limitations. First, state-of-art routing algorithms for SCDC are topology dependent. A specific routing algorithm is only designed for the specific topology (called original routing algorithm).Data center operators have adopted many different network topologies [22] (e.g., folded Clos, FatTrees, VL2, DCell, BCube and so on) [1, 12, 14, 16, 24]. For example, the original routing algorithm for DCell cannot work on other SCDC topologies like BCube. Second, all the original routing algorithms use hop count as routing metric. Without taking path quality into consideration, routing algorithms cannot guarantee satisfied performance for diverse applications in DCNs.

To effectively utilize high performance routing path, general multi-constrained QoS routing algorithms are widely investigated in the context of traditional network, which control traffic of the whole network by adjusting the number of flows through routers and switches, e.g., Multi-Constrained Path problem (MCP) [22], Multi-Constrained Shortest Path (MCSP) [28], Multi-Constrained Optimal Path problem (MCOP) [7], etc. However, all these solutions only calculate one path for an original destination pair and only focus on routings for routers. In addition, these algorithms originally designed for arbitrary topology and no optimizations are considered leveraging the topology characteristics of SCDC. In fact, the performance of the network routing can be significantly improved by exploiting the unique features of SCDC.

In order to overcome the aforementioned limitations and fully utilize multiple paths in SCDC, this paper aims to solve the so-called Multi-Constrained Multi-Path Problem (MCMP), which finds out an optimal path and other sub-optimal paths for routing under multi-constraints. To the best of our knowledge, it is the first time to introduce this problem in the context of SCDC to find optimal routing paths, which is different from finding only multiple paths in DCNs done by previous work. In this paper, we propose Server-Centric Multi-Constrained Routing AlgoriThm (SCRAT) to solve the MCMP problem, which finds feasible paths from source to destination under multi-constraints simultaneously. We utilize the characteristics of SCDC to decrease the complexity of algorithm and propose a specific Multi-Constrained QoS Routing method to weight the cost of links in searching optional paths. Simulations have demonstrated that SCRAT performs much better than original routing algorithm in SCDC.

Specifically, the proposed SCRAT has the following technical merits.

First, SCRAT uses multi-constraints to find out multiple paths with high quality simultaneously. As a result, SCRAT provides an efficient methodology to solve MCMP problem, which is crucial to spread traffic in SCDC. And multiple paths found in SCRAT ensure better available bandwidth and server-to-server throughput than the existing multi-path routing algorithms.

Second, we take topology characteristics of SCDC into consideration when we design SCRAT. As a result, SCRAT makes the relaxation progress more efficiently and decreases the complexity of algorithm and increases the algorithm's accuracy. Compared with present MCOP algorithms that ignore the characteristics of SCDC, SCRAT has a better performance to find the global optimal paths.

Third, SCRAT is a general method for SCDC and can be employed for all the SCDCs topologies, which is an advance over the original SCDC routing algorithms only working for a specific topology.

The rest of this paper is organized as follows. Section 2 overviews existing routing algorithms in Server-Centric network and Multi-Constrained algorithms. Section 3 works out the way to calculate weight vector and cost, and lists some definitions that will be used in this paper. Section 4 describes our SCRAT. Section 5 gives two sets of simulations to evaluate the performance of algorithm. Finally, Section 6 concludes the paper.

2. Related Work

Server-Centric Data Center Network is widely researched around the world. In [2], authors mentioned 5 main disadvantages in traditional data center network, such as no performance isolation, limited management flexibility etc. And the solutions to all of those issues are crucial to the future development of data center. In order to overcome them, many new network architectures had been proposed along with high efficient routing algorithms. For example: BCube original routing algorithm assigns server addresses according to their position characteristics. This algorithm systematically finds intermediate servers by 'correcting' one digit of previous server address [15]. However, as mentioned before, original server-centric algorithm only works well in unique architecture and almost all of those initial algorithms choose the path according to hop count. Some researches have been done on routing in all kinds of DCs to provide multi-paths [23,27]. However, in [27] the first way is spreading load by choosing path randomly. It is obviously not an effective solution. The article also mentioned another way to find multi-paths using multi-static VLANs. The minimal number of VLANs of this solution exponentially depends on the number of equipment in data center. Setting too much VLANs in data center is very expensive. In [23], the multi-path selecting algorithm in SPAIN is running the shortest path algorithm for k times. However, computing shortest paths just consider the hop-count constraint, which cannot ensure good performance of chosen paths. And in the process of repeating shortest algorithm for several times, a large amount of computations are unnecessary. Multi-path routing algorithm can be designed in a much more efficient way.

In the field of routing in a general network, multi-constrained routing problem is widely researched [6,7,19,21,26,29,30]. In order to solve Multi-Constrained QoS routing problems, many algorithms have been proposed. MCP focuses on finding one alternative path. So path selected by this kind of algorithms is just feasible path. MCSP devotes to finding the shortest path under multi-constraints. So it may not balance the cost of network and take full advantage of resources. MCOP is trying to solve the problem that finding the optimal path under multi-constraints. And MCOP problem is the most meaningful problem in this set of issues. In 2002, Korkmaz and Krunz put forward the H_MCOP algorithm [21]. This algorithm has better performance than all previous multi-constraint algorithms. And it can find out feasible path at a very high possibility. Then this kind of problem appealed many people's attention. There are several algorithms improving the performance of H_MCOP, such as TS_MCOP [8], and EH_MCOP [30]. The TS_MCOP was proposed [8], which improved H_MCOP best. Those algorithms work well in finding optimal path. But they do not provide multi-paths. So they cannot be used directly in the context of Server-Centric Network.

In all, present multi-path algorithms in data center are not efficient. And all general multi-constrained routing algorithms do not consider topology characteristics in SCDC and can only provide one feasible path. Neither of them can solve the MCMP problem, which is fairly significant for the performance of SCDC. So we propose a new routing algorithm to solve it.

3. Foundation And Definitions

3.1. Weight and Cost

According to the different characteristics and properties of constraints, they can be divided into the following three categories [10]: additive constraint (e.g. delay, jitter, cost and hop count) multiplicative constraint, (e.g. link reliability and packet loss probability) and concave constraint (e.g. bandwidth). Assume path P has j hops and $w_i(e)$ means the i^{th} weight of edge e. According to the constraints, we proposed the following functions to compute weights of a path:

Additive constraint The weight of additive constraint is represented by summing every link's weight together, shown in (1).

$$w_i(P) = \sum_{l=1}^{j} w_i(e_l) \tag{1}$$

where i is the serial number of all additive constraints.

Multiplicative constraint By converting logarithm form of multiplicative constraints to additive constraints , the weight can be calculated with (2)

$$w_i(P) = \prod_{l=1}^{j} w_i(e_l) = e^{(\sum_{l=1}^{n} \ln[w_i(e_l)])} \tag{2}$$

where i is the serial number of all multiplicative constraints.

Concave constraint concave constraints mark the limit of path, and can be directly used as boundaries for selecting paths. So, (3) is used to calculate those concave weights.

$$w_i(P) = min\{w_i(e_1), w_i(e_2), ..., w_i(e_j)\} \tag{3}$$

where i is the serial number of all concave constraints.

In order to calculate various constraints in one function, Jaffe [19] used the linear cost function to represent the cost of path, shown in (4)

$$COST(P) = \sum_{i=1}^{k} d_i w_i(P) \tag{4}$$

where $COST(P)$ indicates the cost of path P and d_i is the coefficient of w_i.

This representing method can be used to calculate the cost of path for Dijkstra Algorithm, which are utilized by many former routing algorithms. However, linear function cannot reflect the real constraints very well. In order to fit actual constraints better, nonlinear function (5) was proposed to calculate the cost of path [6].

$$COST(P) = [\sum_{i=1}^{k} [\frac{w_i}{c_i}]^q]^{\frac{1}{q}} \tag{5}$$

when $q \to \infty$

$$COST_\infty(P) = \max_{1 \leq i \leq k} [\frac{w_i(P)}{C_i}] \tag{6}$$

By (6), we can precisely find out all feasible paths that meet the multi-constrained requirements. If we use (6) to calculate the cost of paths, Dijkstra algorithm is not suitable any more. So we need to work out an algorithm that can calculate cost with nonlinear function.

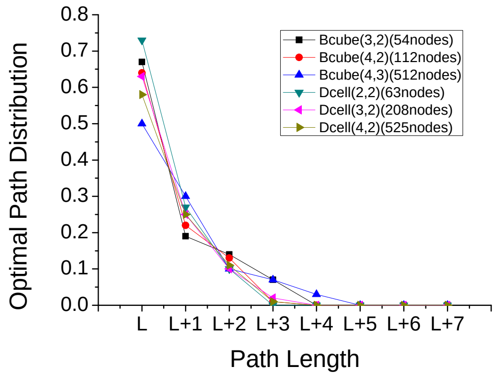
3.2. Definitions

Definition 1 *Server-Centric Data Center*: in server-centric data center, servers act not only as end hosts but also as relay nodes for multi-hop communications. [4] In SCDC, there are links that connect servers directly and there is no traditional hierarchic switch structure, which may cause bottleneck in whole network. In SCDC each server links to several, not one, servers or switches, which balances the load of overall network greatly.

Definition 2 *Feasible Path*: given a weighted network graph $G(V, E)$, where V represents the set of nodes and E represents the set of edges, $n = |V|$ and $m = |E|$. Each edge $e(v_i, v_j)$ has a link weight vector W with components of K link weight $w_k \geq 0$ for all $1 \leq k \leq K$. And the corresponding constraint vector C with K constraints c_k. A path is a sequence with non-repeated nodes $P = (v_1, v_2, ..., v_i)$. Since there are different types of constraints, simply adding weights together is unreasonable. A feasible path $P = (v_1, v_2, ..., v_i)$ so that $w_k(P) \leq c_k$ for all $1 \leq k \leq K$.

Definition 3 *Optimal path*: in all feasible paths from v_i to v_j noting as $P_1, P_2, ..., P_l$, we use (5) to calculate the cost of paths. Then the optimal path is the path P_0 satisfied: $COST(P_0) \leq COST(P_i)$ for all $1 \leq i \leq l$.

Definition 4 *Neighbor node pair*: in Server-Centric network, if two servers v_i and v_j are linked directly or they interconnect each other through one switch, (v_i, v_j) are neighbor node pair. If two servers interconnect via another server, they cannot be regarded as neighbor nodes.



ss

Fig. 1. The distribution of optimal path's path-length in SCDC.

Definition 5 *Neighbor node matrix*: in a Server-Centric network with N servers, the neighbor node matrix is a N^2 matrix M_1. Each element $v_{i,j}$ in M_1 contains hop count, weight vector and cost of the neighbor node path that links v_i and v_j together. If v_i and v_j are neighbor node pair, we note down hop count, weight vector and cost in $v_{i,j}$. If v_i and v_j are not neighbor node pair, we note 0 in $v_{i,j}$. If v_i and v_j connect directly, $hop\ count = 1$. If v_i and v_j are connected by a switch, then $hop\ count = 2$.

Definition 6 *Path-length*: in this paper, if v_i and v_j are neighbor node pair, we define the path-length of path (v_i, v_j) to be 1. If v_i and v_j are neighbor node pair and v_j and v_k are neighbor node pair, then there is a path (v_i, v_j, v_k) between (v_i, v_k), and the path-length of this path is 2. Paths with path-length 3, 4 and so on can be defined similarly.

4. Algorithm Design

In SCDC, all switches are connected to servers directly. If two servers are neighbor node pair but do not connect directly, it is easy to figure out the intermediate switch's ID through the IDs of those two servers. This kind of topology characteristic offers us great favor to simplify our searching strategy. So when designing routing algorithm in Server-Centric network, we should pay more attention on servers instead of switches and find an efficient way to route by servers. If we fully employ the topology characteristic in SCDC, the routing algorithm can be simplified and more efficient.

Furthermore, as mentioned in the definition of SCDC, no hierarchical structure in SCDC and the linkage is quite flexible. So there are more than one shortest path. And the number of paths own the same path-length of shortest path is even larger. As we know, in general network when the load in overall network is balanced, the shortest path is the optimal path. And in SCDC, due to those topology characteristics of SCDC, the path length of optimal path is very close to the path length of shortest path. We do some research on most widely used SCDC topologies(BCube and DCell). In Figure 1, we can see that when one feasible path's path-length is bigger than $L + k + 1$, where L represents

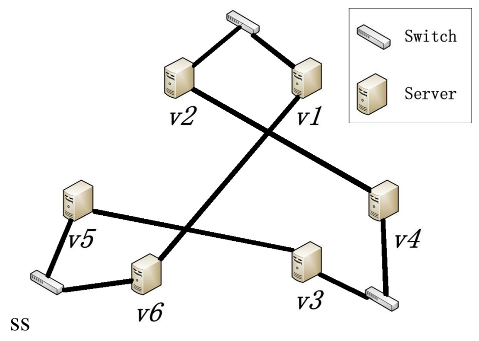


Fig. 2. DCell(2,1)

the path-length of shortest path, it owns fairly low prolixity to become optimal path. This feature guides us to work out a routing algorithm based on the increase of path-length, which is more efficient for SCDC network.

4.1. Algorithm Description

The basic idea of SCRAT is using paths of path-length 1 and paths of path-length N to find paths of path-length $N+1$. SCRAT takes advantage of the fundamental idea of Warshall algorithm to search alternative paths in network. Warshall algorithm is a high-efficiency algorithm to work out the transitive closure of binary relation. However, this algorithm itself can only judge the connectivity of any two nodes in a network, and the complexity of this algorithm is high. In our design, all feasible paths are stored while searching in the graph and the time complexity is decreased successfully by applying the topology characteristics of SCDC.

In order to make the algorithm easily to understand, firstly we use a small network as an example to depict it. In Figure 2, we build a small DCell model (n=2,k=1) as an example. There are six severs and three switches in this network. And we use four constraints $[c_1, c_2, c_3, c_4]$, in which c_1, c_2 are additive constraints (e.g. hop count; delay); c_3 is multiplicative constraint(e.g. link reliability) and c_4 is concave constraint (e.g. bandwidth). So there are four corresponding weights for any path P. We note the weight vector of P as

$$W_P = [w_1(P), w_2(P), w_3(P), w_4(P)]^T \tag{7}$$

Considering any two servers $v_i, v_j (1 \leq i \leq 6, 1 \leq j \leq 6)$, if (v_i, v_j) is neighbor node pair, then we can calculate the cost of this path with (5). Then we store hop count and other related information (RI): weight vector and cost value in $v_{i,j}$ of M_1. The result is shown in TABLE I (a).

Then we use M_1 to build another vector M_2 that records all feasible paths with path-length 2 and their corresponding information: hop count, weight vector, media servers and cost. Use v_1 as an example. To get paths with the length of two, we first search all neighbor servers of v_1 in M_1. Hence we get v_2 and v_6. We can directly arrive at v_1, v_4 from v_2 and v_1, v_5 from v_6. Removing reduplicative paths and nodes, we get two paths from v_1 with path-length 2: (v_1, v_2, v_4) and (v_1, v_6, v_5). Calculate the weight vectors of

Table 1.

(a) Neighbor Node Matrix

M_1	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	$\overset{2}{RI}$	0	0	0	$\overset{1}{RI}$
v_2	$\overset{2}{RI}$	0	0	$\overset{1}{RI}$	0	0
v_3	0	0	0	$\overset{2}{RI}$	$\overset{1}{RI}$	0
v_4	0	$\overset{1}{RI}$	$\overset{2}{RI}$	0	0	0
v_5	0	0	$\overset{1}{RI}$	0	0	$\overset{2}{RI}$
v_6	$\overset{1}{RI}$	0	0	0	$\overset{2}{RI}$	0

(b) Path-Length 2 Matrix

M_2	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	0	0	$\overset{3\ v_2}{RI}$	$\overset{3\ v_6}{RI}$	0
v_2	0	0	$\overset{3\ v_4}{RI}$	0	0	$\overset{3\ v_1}{RI}$
v_3	0	$\overset{3\ v_4}{RI}$	0	0	0	$\overset{3\ v_5}{RI}$
v_4	$\overset{3\ v_2}{RI}$	0	0	0	$\overset{3\ v_3}{RI}$	0
v_5	$\overset{3\ v_6}{RI}$	0	0	$\overset{3\ v_3}{RI}$	0	0
v_6	0	$\overset{3\ v_1}{RI}$	$\overset{3\ v_5}{RI}$	0	0	0

where RI (related information) contains weight vector and cost value.

the two paths using (1) (2) (3), then calculate cost by (5). Here we use path (v_1, v_2, v_4) as an example. First two constraints c_1, c_2 are additive constraints, so we choose (1) to compute the weights of first two constraints. So

$$w_1(v_1, v_2, v_4) = w_1(v_1, v_2) + w_1(v_2, v_4) \tag{8}$$

$$w_2(v_1, v_2, v_4) = w_2(v_1, v_2) + w_2(v_2, v_4) \tag{9}$$

The third constraint c_3 is multiplicative constraint, so corresponding weight $w_3(P)$ should use (2) to compute:

$$w_3(v_1, v_2, v_4) = e^{ln(w_3(v_1,v_2))+ln(w_3(w_2,w_4))} \tag{10}$$

Forth constraint is concave constraint, use (3) to calculate the corresponding weight:

$$w_4(v_1, v_2, v_4) = min\{w_4(v_1, v_2), w_4(v_2, v_4)\} \tag{11}$$

The weight vector of path (v_1, v_2, v_4) can be present in the following form:

$$W_{(v_1,v_2,v_4)} = [w_1(v_1v_2v_4), w_2(v_1v_2v_4), w_3(v_1v_2v_4), w_4(v_1v_2v_4)]^T \tag{12}$$

Comparing this weight vector with constraints, if all weights meet multi-constrained requirements, we compute the cost of this path:

$$COST_{(v_1,v_2,v_4)} = \{\sum_{i=1}^{4}[\frac{w_i(v_1, v_2, v_4)}{c_i}]^q\}^{\frac{1}{q}} \tag{13}$$

If any weight of this path excesses the required limitation, we drop this path off. Because all weights are increasing with the increase of hop count, so if a path P_i cannot meet multi-constrained requirement, any other path with a sub-path P_i also cannot meet requirement, too.

In this example, assuming that all available paths meet requirements, we record those paths and its relevant information (hop count, intermediate server, weight vector and cost) into matrix M_2. Similarly, other paths can be calculated in this way, then M_2 is built shown in TABLE I (b).

In a similar fashion, with the information from M_1 and M_2, we can build M_3. Then M_4, M_5,...,M_x can be built. Where x indicates the maximum number of path-length that is limited by the QoS requirements. Since the topology of SCDC is very efficient, x is always a small number. For example, in BCube, x can be set as $l+1$ (l is the port number

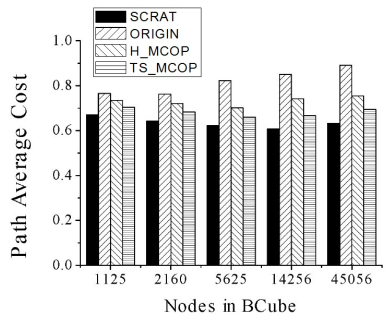
```
SCRAT

1:  for $(u = 1; u \leq x; u++)$ do
2:    for $(i = 1; i \leq N; i++)$ do
3:      if $u == 1$ then
4:        $Find\ all\ neighbor\ nodes\ \{v_{j1}, v_{j2}, ...\}\ of\ v_i$
           $from\ its\ ID$
5:        if $(v_i, v_{jk})$ meets all constraints then
6:          $Calculate\_COST(v_i, v_j)$
7:          $Add[v_i, v_j] into\ Matrix[u, i, j]$
8:        end if
9:      else
10:       $A = Getsort(v_i)$
11:       $v_y = any\ node\ in\ A$
12:       if $[v_y, ..., v_j] is\ in\ Matrix[u-1, i, j]$ then
13:         if $(v_i, v_y, ..., v_j) meets\ all\ constraints\ \&\ No\ circle$ then
14:           $Calculate\_COST(v_i, v_y, ..., v_j)$
15:           $Add[v_i, v_y, ..., v_j] into\ Matrix[u, i, j]$
16:         end if
17:       end if
18:     end if
19:   end for
20: end for
21: $Sort\ all\ those\ available\ links\ according\ to\ cost$
```

Fig. 3. Server-Centric Multi-Constrained Routing Algorithm pseudo-code.

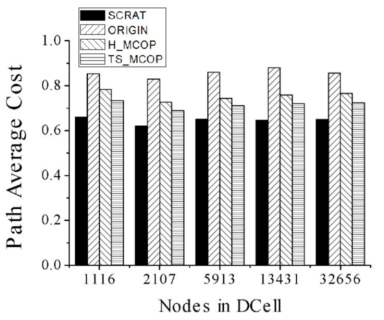
of a server). All feasible paths between any two servers are available from those tables. We sort them according to their cost, then we can find the optimal path and other alternative paths as well.

The general algorithm is shown in Figure 3. In the algorithm, $Matrix[u, i, j]$ records all paths and their weigh vectors and costs from v_i to v_j with path-length u. The loop in first line is to search all paths with path-length less than x. The 10^{th} line is to pick up all neighbor nodes of v_i. 6^{th} and 14^{th} lines are calculating the cost of paths using (5). 21^{th} line sorts all feasible paths according to paths' cost. Then we can get the optimal path and other sub-optimal paths as well.



(a) BCube scale increases from 1125nodes (BCube(5,3)) to 45056 nodes (BCube(4,6))

ss



(b) DCell scale increases from 1116nodes (DCell(5,2)) to 32656 nodes (DCell(3,3))

Fig. 4. The average cost of optimal path selected by different algorithms in different scale of BCube and DCell.

4.2. Complexity of Algorithm

SCRAT is an all-to-all routing algorithm. Time complexity of SCRAT is $O((k-1)^x N)$, where N represents the number of servers; k represents the number of ports on a server and x indicates the given max limitation of path-length. Both k and x are small constants compared with N. The searching cost of prior one matrix is N^2. All M_2,M_3,...,M_x matrix needed to be calculated, so we need to repeat $x-1$ times. And for each possible path, we need to check all its neighbors, the cost is $k-1$. In any matrix M_i, for any two nodes, the average number of possible paths is less than $(k-1)^{i-1}/N$. So the total complexity is $\sum_{i=2}^{x} k(k-1)^{i-1}N \leq k[(k-1)^x-(k-1)]/(k-2)*N = O((k-1)^x N)$. Meanwhile we need a matrix when we store all paths in each path-length. So the spatial complexity of SCRAT is $O(x*N^2)$.

4.3. Proof of Optimality

In this part, we will prove that if there exists an optimal path satisfying the multi-constrained requirements, SCRAT can guarantee to find it.

Assume that the source server is v_i, and the destination server is v_j. And there is an optimal path meeting multi-constrained requirements, noting it as $(v_i, v_{m1}, v_{m2}, ..., v_{mn}, v_j)$. So this optimal path's sub-paths$(v_{m1}, v_{m2}, ..., v_{mn}, v_j)$,$(v_{m2}, ..., v_{mn}, v_j)$,$(v_{m3}, ..., v_{mn}, v_j)$,..., (v_{mn}, v_j) all meet the multi-constrained requirements. Then node pairs(v_i, v_{m1}), (v_{m1}, v_{m2}),...,(v_{mn}, v_j) are all in neighbor node matrix M_1. For the reason that $(v_{m,n-1}, v_{mn})$ and (v_{mn}, v_j) are in M_1,$(v_{m,n-1}, v_{mn}, v_j)$ is in M_2. Due to $(v_{m,n-2}, v_{m,n-1})$ is in M_1, $(v_{m,n-2}, v_{m,n-1}, v_{mn}, v_j)$ is in M_3. And so on in a similar fashion, the path $(v_i, v_{m1}, v_{m2}, ... , v_{mn}, v_j)$ must be in the matrix M_{n+1}. So SCRAT can guarantee us to find the optimal path if it exists.

Table 2.

(a) BCube(n,k) Nodes Number

(n,k)	$Nodes$	(n,k)	$Nodes$
$(3,2)$	54	$(6,3)$	2160
$(4,2)$	112	$(5,4)$	5625
$(4,3)$	512	$(6,4)$	14256
$(5,3)$	1125	$(4,6)$	45056

(b) DCell(n,k) Nodes Number

(n,k)	$Nodes$	(n,k)	$Nodes$
$(2,2)$	63	$(6,2)$	2107
$(3,2)$	208	$(8,2)$	5913
$(4,2)$	525	$(10,2)$	13431
$(5,2)$	1116	$(3,3)$	32656

5. Simulation

5.1. Simulation Settings

In simulations, we take most widely researched and used SCDC topologies: BCube [13] and DCell [15] as our architectures. The algorithms' performances in other SCDCs are similar. The scale of those two types of topologies is shown in TABLE II. Four metrics are selected as multi-constraints: hop count, delay, package loss probability and bandwidth. The hop count between any two servers is fixed due to the structure of network. The bandwidth of all links in topologies are 1Gb. The initial values of other three metrics are assigned randomly. Specifically the original values of delay, containing waiting time in node and transforming time on link, obey uniform distribution in the interval $(0, 200)$us. Values of package loss probability obey uniform distribution in the interval $(0\%, 5\%)$. The initial values of used bandwidth obey uniform distribution in the interval $(0, 0.6)$Gb. The constrains generate by $1.5w_k(p)$, where p is the shortest path from source to destination [21]. In each simulation, source and destination are selected randomly and each simulation is repeated for 500 times.

We build two sets of simulations to evaluate the performance of SCRAT. The first set of simulations are to evaluate the quality of the optimal path in SMCMRA. Although SCRAT is a multi-path algorithm, it is also quite important to guarantee the selected optimal path has a high quality. And the second set of simulations are to evaluate the performance of selected multi-paths in SCRAT.

5.2. Simulation Results

– Optimal Path Simulations

Average Path Cost The first simulation compares the optimal path found in SCRAT with the original algorithm, TS_MCOP and H_MCOP. We run each algorithm on different scales of BCube and DCell respectively. And we calculate the cost of path selected in original algorithms, so we can compare them directly. Figure 4 shows the average cost of optimal path found in different algorithms. In those two figures, SCRAT can decrease path cost about 10% compared with TS_MCOP.

In second simulation, we compare the optimality, which is the probability to find the optimal solution if there exists at least one feasible path [8], of SCRAT, H_MCOP and TS_MCOP. Because computing of exact optimal path under multi-constraints is an

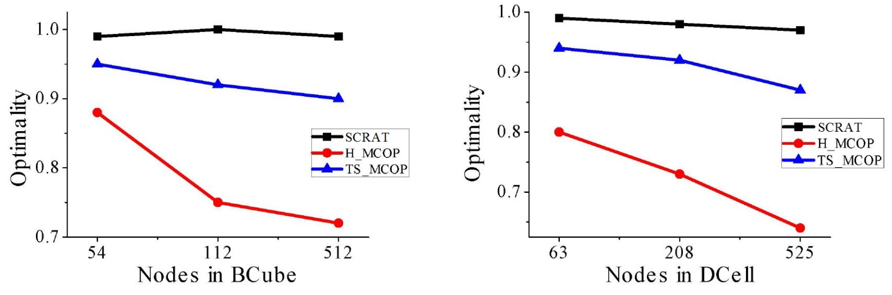
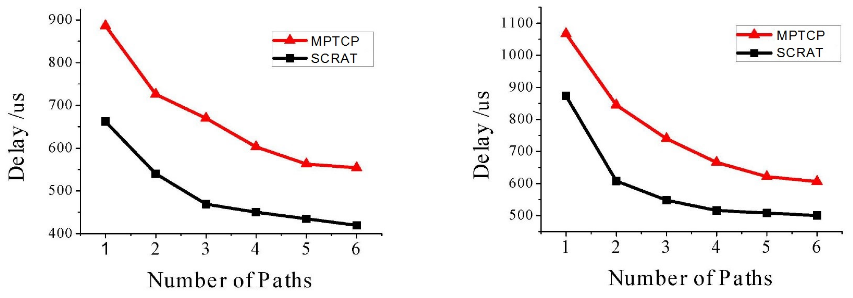


Fig. 5. Optimality is the probability that one algorithm find the exact optimal path under multi-constraint. Because finding the exact optimal path under multi-constraint is an NP problem, we just calculate the exact optimal path in small scale of BCube and DCell.



(a) BCube(6,4) 14256 nodes (b) DCell(10,2) 13431 nodes

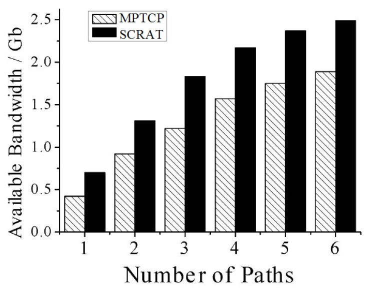
Fig. 6. The delay in DCell and BCube using multi-paths computed by SCRAT and MPTCP respectively.

NP problem and it is almost impossible to work out exact optimal path in large scale of topology, we use small scale of BCube and DCell in simulation. Figure 5 shows the results, where optimality reflects the possibility that the chosen path is the exact optimal path. We can find that SCRAT has the largest possibility to find out the exact optimal path than other algorithms. And with the increase of topology, optimality decreases in a very low rate. So it is convincing that in large scale of topology, SCRAT performs well in finding the exact optimal path.

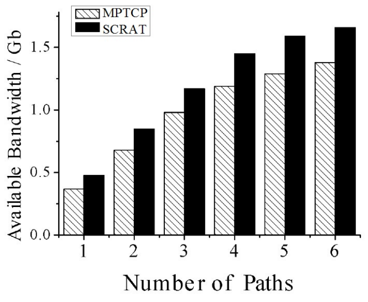
Optimality The second set of simulations compare the performance of multi-paths in SCRAT with MPTCP.

– Multi-Path Simulations

Delay of Multi-Path When we divide one flow into several sub-flows, the overall delay of multiple paths is the maximum delay of all paths. Using multiple paths, the time consumed to pass through links decreases little, but the waiting and processing time in nodes decreases significantly. Figure 6 compares the overall delay of different number of paths selected by SCRAT and MPTCP respectively in BCube and DCell. From the figure, we can find that multiple paths can efficiently decrease the delay to transfer flow. When the number of paths is small, increasing one more path can decrease delay apparently. Compared with paths found in MPTCP, multiple paths found in SCRAT can decrease delay at least 18%.



(a) BCube(6,4) 14256 nodes



(b) DCell(10,2) 13431 nodes

Fig. 7. The available bandwidth in BCube and DCell using multi-paths computed by SCRAT and MPTCP respectively.

Available Bandwidth of Multi-Path Figure 7 compares the available bandwidth of different number of paths selected by SCRAT and MPTCP respectively. From the figure, we can find that multi-paths found in SCRAT achieve 20% more available bandwidth than MPTCP. And with the increase of selected paths, available bandwidth in SCRAT grows faster than MPTCP.

6. Conclusion

MCMP is a very important problem for efficient traffic spreading in SCDC, which has not been solved previously. This paper propose SCRAT to solve the MCMP problem, which leverages the topology characteristics of Server-Centric data center. The algorithm decreases the complexity of the algorithm and simplifies the routing process. Given the path-length, SCRAT can find the optimal path and other sub-optimal paths under multi-constraint. Simulations demonstrate that SCRAT has a very large possibility to find out the exact optimal path and path cost is also lower than the optimal path cost in other multi-constraint algorithm. Additionally, multiple paths found in SCRAT can decrease delay and increase available bandwidth compared with MPTCP.

Acknowledgments. This research was supported by the National Natural Science Foundation of China (Grant No. 61170245, No. 61272459,and No. 61221063), and the Open Project of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory of China (No.ITD-U14001/KX142600008).

References

1. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. ACM SIGCOMM Computer Communication Review 38(4), 63–74 (2008)
2. Bari, M.F., Boutaba, R., Esteves, R., Granville, L.Z., Podlesny, M., Rabbani, M.G., Zhang, Q., Zhani, M.F.: Data center network virtualization: A survey. Communications Surveys & Tutorials, IEEE 15(2), 909–928 (2013)
3. Benson, T., Anand, A., Akella, A., Zhang, M.: Understanding data center traffic characteristics. ACM SIGCOMM Computer Communication Review 40(1), 92–99 (2010)
4. Chen, K., Hu, C., Zhang, X., Zheng, K., Chen, Y., Vasilakos, A.V.: Survey on routing in data centers: insights and future directions. Network, IEEE 25(4), 6–10 (2011)
5. Chen, K., Singla, A., Singh, A., Ramachandran, K., Xu, L., Zhang, Y., Wen, X., Chen, Y.: Osa: an optical switching architecture for data center networks with unprecedented flexibility. Networking, IEEE/ACM Transactions on 22(2), 498–511 (2014)
6. Dai, F.S., Liu, A.J.: A multi-constrained quality of service routing algorithm based on vector converting. In: Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on. pp. 1–4. IEEE (2009)
7. De Neve, H., Van Mieghem, P.: Tamcra: a tunable accuracy multiple constraints routing algorithm. Computer Communications 23(7), 667–679 (2000)
8. Fang, Q., Han, J., Mao, L., Li, Z.: Exact and heuristic algorithm for multi-constrained optimal path problem. In: Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. pp. 45–51. IEEE (2011)
9. Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H.H., Subramanya, V., Fainman, Y., Papen, G., Vahdat, A.: Helios: a hybrid electrical/optical switch architecture for modular data centers. ACM SIGCOMM Computer Communication Review 41(4), 339–350 (2011)
10. Fu, Y., Cheng, X., Tang, Y.: Optimization theory and method. Press of UESTC (1996)
11. Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S.: Vl2: a scalable and flexible data center network. In: ACM SIGCOMM computer communication review. vol. 39, pp. 51–62. ACM (2009)
12. Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S.: Vl2: a scalable and flexible data center network. In: ACM SIGCOMM computer communication review. vol. 39, pp. 51–62. ACM (2009)
13. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review 39(4), 63–74 (2009)
14. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review 39(4), 63–74 (2009)
15. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. ACM SIGCOMM Computer Communication Review 38(4), 75–86 (2008)
16. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. ACM SIGCOMM Computer Communication Review 38(4), 75–86 (2008)
17. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: Elastictree: Saving energy in data center networks. In: NSDI. vol. 10, pp. 249–264 (2010)
18. Hong, C.Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., Wattenhofer, R.: Achieving high utilization with software-driven wan. In: ACM SIGCOMM Computer Communication Review. vol. 43, pp. 15–26. ACM (2013)
19. Jaffe, J.M.: Algorithms for finding paths with multiple constraints. Networks 14(1), 95–116 (1984)

20. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., et al.: B4: Experience with a globally-deployed software defined wan. In: ACM SIGCOMM Computer Communication Review. vol. 43, pp. 3–14. ACM (2013)
21. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. vol. 2, pp. 834–843. IEEE (2001)
22. Kuipers, F., Van Mieghem, P., Korkmaz, T., Krunz, M.: An overview of constraint-based path selection algorithms for qos routing. IEEE Communications Magazine, 40 (12) (2002)
23. Mudigonda, J., Yalagandula, P., Al-Fares, M., Mogul, J.C.: Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In: NSDI. pp. 265–280 (2010)
24. Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: Portland: a scalable fault-tolerant layer 2 data center network fabric. In: ACM SIGCOMM Computer Communication Review. vol. 39, pp. 39–50. ACM (2009)
25. Porter, G., Strong, R., Farrington, N., Forencich, A., Chen-Sun, P., Rosing, T., Fainman, Y., Papen, G., Vahdat, A.: Integrating microsecond circuit switching into the data center, vol. 43. ACM (2013)
26. Puri, A., Tripakis, S.: Algorithms for the multi-constrained routing problem. In: Algorithm TheorySWAT 2002, pp. 338–347. Springer (2002)
27. Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., Handley, M.: Improving datacenter performance and robustness with multipath tcp. ACM SIGCOMM Computer Communication Review 41(4), 266–277 (2011)
28. Reinhardt, L.B., Pisinger, D.: Multi-objective and multi-constrained non-additive shortest path problems. Computers & Operations Research 38(3), 605–616 (2011)
29. Van Mieghem, P., Kuipers, F., et al.: Concepts of exact qos routing algorithms. Networking, IEEE/ACM Transactions on 12(5), 851–864 (2004)
30. Wang, S., Wang, H., Li, L.: An enhanced algorithm for multiple constraints optimal path calculation. In: Communications, Circuits and Systems, 2004. ICCCAS 2004. 2004 International Conference on. vol. 1, pp. 703–713. IEEE (2004)

Huanzhao Wang received her B.S., M.S., and Ph.D. degrees in Computer Science and Technology from Xi'an Jiaotong University, China. She is currently an Associate Professor of Department of Computer Science and Technology at Xi'an Jiaotong University. Her research interests include wireless sensor networks and Software Defined Networks.

Kun Qian received his B.S. degree in Computer Science and Technology from Xi'an Jiaotong University, China, in 2015. He is currently working toward the Ph.D. degree at the Department of Computer Science and Technology at Tsinghua University, Beijing, China. His research interests include network measurement and monitoring.

Chengchen Hu received the B.S. degree from Northwestern Polytechnical University, Xi'an, China, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2003 and 2008, respectively. From June 2008 to December 2010, he worked as an Assistant Research Professor with Tsinghua University. He is currently a Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China. His main research interests include computer networking systems, and network measurement and monitoring. He severed in the organization committee and technical program committee of several conferences.

Che Zhang received her B.S. and M.S. degrees in computer science and automation respectively from Xi'an Jiaotong University, China. She is a PhD candidate in computer science of City University of Hong Kong. Her research interests include SDN, cloud computing, cloud robot, network, algorithm, lego and MIT App Inventor.

Yadong Zhou received his B.S., M.S., and Ph.D. degrees in control science and technology from Xi'an Jiaotong University, China. He is currently an assistant professor of department of automation at Xi'an Jiaotong University. His research interests include software defined networks, online social networks.

Received: February 28, 2016; Accepted: May 17, 2016.